Imbalanced RankBoost for Efficiently Ranking Large-Scale Image/Video Collections

Michele Merler
Computer Science Department
Columbia University, New York, NY 10027
mmerler@cs.columbia.edu *

Rong Yan, John R. Smith
IBM T.J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
{yanr,jsmith}@us.ibm.com †

Abstract

Ranking large scale image and video collections usually expects higher accuracy on top ranked data, while tolerates lower accuracy on bottom ranked ones. In view of this, we propose a rank learning algorithm, called Imbalanced RankBoost, which merges RankBoost and iterative thresholding into a unified loss optimization framework. The proposed approach provides a more efficient ranking process by iteratively identifying a cutoff threshold in each boosting iteration, and automatically truncating ranking feature computation for the data ranked below. Experiments on the TRECVID 2007 high-level feature benchmark show that the proposed approach outperforms RankBoost in terms of both ranking effectiveness and efficiency. It achieves an up to 21% *improvement in terms of mean average precision, or equivalently, a 6-fold speedup in the ranking process.*

1. Introduction

Recent years have seen unprecedented growth of image and video data. The cases of Youtube[1] (13 hours of video uploaded every minute) and Flickr[2] (5000 images uploaded per minute) are emblematic. Such a growth has led to a greater need for developing efficient algorithms for ranking and retrieving large-scale image/video collections. Learning to rank, which aims to automatically identify better ranking function using historical data, has become increasingly important in this domain.

The success of rank learning have been demonstrated in many other applications, such as online search engines, collaborative filtering and recommendation systems. Numerous learning-to-rank algorithms have been proposed before, such as RankBoost[6, 15, 2], active learning[10], ranking logistic regression[17], RankSVM[4]. However, none of them account for the fact that, a vast majority of users prefer to browse only a limited number of top ranked examples [12], while completely ignore the rest of them. Without any distinctions among ranked results, existing methods is prone to allocate a considerable amount of computational resources to less visible bottom-ranked examples, and thus lead to an inefficient ranking process. For large scale collections, it is desirable to develop a ranking algorithm that has more emphasis on the efficiency and accuracy of top examples rather than bottom ones.

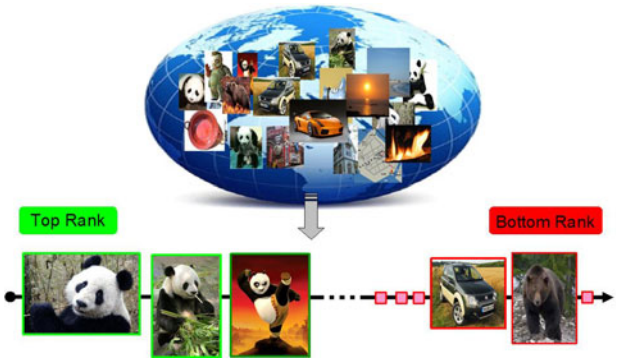

Figure 1. Example of ranking application which ranks an unordered image collection (top) into a sorted list of images (bottom). Most general users spend more time in examining the top ranked examples on the left, while ignoring the bottom-ranked ones on the right.

To this end, we propose a rank learning algorithm called Imbalanced RankBoost by integrating the principles of RankBoost with an iterative thresholding scheme. In addition to iteratively selecting and combining weak ranking features, the proposed approach provides a more efficient ranking process by automatically identifying a cutoff threshold in each run, and truncating model computation for the examples ranked below. A unified loss optimiza-

*This work was supported in part by the National Science Foundation (NSF) under grant IIS-07-13064.

†This material is based upon work supported by the US Government. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Government.

[1] http://googleblog.blogspot.com/2008/09/future-of-online-video.html
[2] http://www.flickr.com/photos/

tion framework is provided to jointly learn the ranking features, cut-off thresholds and combination weights. Our experiments on the TRECVID 2007 high-level feature detection benchmark confirms that the proposed approach outperforms RankBoost in terms of both ranking effectiveness and efficiency. In more details, it achieves either up to 21% performance improvement in terms of mean average precision with the same amount of ranking time, or equivalently, a 6-fold speedup when they reach the same level of ranking performance.

The rest of the paper is organized as follows: we present in Section 2 relevant work on the ranking problem, Section 3 reviews the RankBoost algorithm, Section 4 introduces and discusses the proposed Imbalanced RankBoost algorithm, and Section 5 presents experimental results on the TRECVID 2007 high level feature extraction benchmark. We conclude and discuss future directions in Section 6.

2. Related Work

Numerous algorithms have been recently proposed to address the rank learning problem. For example, Burges et al. [3] use a neural network to model the underlying pairwise ranking function, and optimize the probabilistic cost with gradient descent. In [10] an active learning approach is introduced, which aims to find the most useful unlabeled data for learning the ranking function. RankBoost [6] automatically selects and combines a pool of weak ranking features into an composite ranking function. It has been successfully applied in computer vision, e.g. [16] for face alignment. In [15] a rejection sampling variant of RankBoost tries to alleviate the computational burden by sampling the training data. Despite the effectiveness of such sampling-based rank learning methods, the entire set of ranking features are unselectively applied to every example, which thus wastes a considerable amount of resources on less important data.

Dedicating more computations to the small fraction of top examples shares resemblance with re-balancing data distribution in imbalanced collections, if we assume top ranked data are mostly relevant. Popular approaches for mitigating the data imbalance issue includes synthetic positive data generation [7], over-sampling relevant data, and down-sampling irrelevant data [5]. More recently, Masnadi-Shirazi et al. [9] introduced a cost sensitive extension of the Adaboost algorithm. Wang et al. [14] balanced the distribution of examples by down sampling the irrelevant data before boosting a pool of soft margin SVMs. Cao et al. [4] proposed a variant of RankSVM, which assigns higher penalties to top examples that are mis-ranked. All the aforementioned algorithms, however, do not take into consideration the issue of learning and prediction efficiency. As an alternative, our proposed work can improve ranking accuracy as well as efficiency by assigning more complex ranking functions to more important examples.

RankBoost

1. Input: training set X, a pool $\mathcal{H}$ of ranking features $h^k(x) \in [0,1]$
2. For $t = 1, \ldots, T$
   (a) Choose a ranking feature $h_t^k(x)$ from $\mathcal{H}$ to optimize exponential ranking loss L in Eq. 1
   (b) Compute α_t to optimize L
   (c) Update ranking function $f_t \leftarrow f_{t-1} + \alpha_t h_t^k$
3. Output: final ranking $F(x) = \sum_{t=1}^{T} \alpha_t h_t^k(x)$

Figure 2. Outline of RankBoost (loss optimization version).

For the task of object detection, Viola and Jones [13] proposed a cascaded classification framework for face detection, which use simple classifiers to quickly filter out most of irrelevant examples at the early stage of cascading process. To some extent, the proposed approach adopts a similar cascading principle in the ranking problem. But in our case, the ranking algorithm is not allowed to discard or prune any examples because of the requirement of complete ordering. Instead, it only allocates less computational resources to the bottom ranked examples, which are still retrievable. Moreover, rather than resorting to heuristics to determine the cascading thresholds, we provide a principled solution by unifying the selection of cutoff thresholds and ranking features into a loss optimization framework.

3. RankBoost

The goal of rank learning consists in producing a ranked list of originally unordered image/video examples, so that the relevant ones are placed as close as possible to the top. Formally, a rank-learning algorithm orders the data by learning a ranking function $F(x) : \mathbb{R}^n \rightarrow \mathbb{R}$, where $F(x_i) > F(x_j)$ means that x_i is ranked higher than x_j. The algorithm takes as input a set of training data $\{x_1, \ldots, x_N\}$ as well as the ranking ground truth expressed as a feedback function $\Phi : X \times X \rightarrow \mathbb{R}$. Such feedback function encodes pair-wise preferences on data examples. Given any two samples x_i and x_j, $\Phi(x_i, x_j) > 0$ if x_i is ranked higher than x_j, $\Phi(x_i, x_j) < 0$ means the opposite, $\Phi(x_i, x_j) = 0$ means no preference between the two samples. However, because it is time-consuming to obtain full pairwise preferences, we typically consider a simplified setting called "bipartite ranking", i.e., X is partitioned into two disjoint subsets where X_0 contains relevant samples and X_1 contains irrelevant ones, such that $\Phi(x_1, x_0) = 1$ $\forall x_0 \in X_0, x_1 \in X_1$, and 0 otherwise.

RankBoost is a rank learning algorithm proposed by Fre-

und et al. [6]. As outlined in Figure 2, it essentially learns to order a set of examples by combining a pool of K "weak" ranking features $h^k(x) \in [0,1]$ into a more accurate composite ranking function $F(x)$. In the bipartite setting, the ranking loss function of RankBoost can be represented as an exponential function on the margin of training data,

$$\begin{aligned} L &= \sum_{x_0,x_1} \Phi(x_0,x_1) \exp\left[-(F(x_0) - F(x_1))\right] \\ &= \sum_{x_0}\sum_{x_1} Z_t C_t(x_0,x_1) \exp\left[\alpha_t h_t^k(x_1) - \alpha_t h_t^k(x_0)\right], \end{aligned} \tag{1}$$

where $C_t(x_0,x_1) = \exp[f_{t-1}(x_1) - f_{t-1}(x_0)]/Z_t$ is a sampling distribution of the training pair between x_0 and x_1, Z_t is the normalization factor. Minimizing the RankBoost loss function is closely related to minimizing the number of ranking errors. This follows from the fact that the RankBoost loss is an upper bound of the misranking error. In order to minimize L, RankBoost iteratively chooses a ranking feature $h_t^k(x)$ from the feature pool, followed by computing a combination weight α_t for the selected ranking feature. The details of the optimization method can be found in [6]. In this work, since weak ranking features can take any real values in the range [-1,1], we approximate the exponential loss function with a linear upper bound using inequality $2e^{-\alpha x} < (1+x)e^{-\alpha} + (1-x)e^{\alpha}$. This upper bound can be minimized in a closed form for a given ranking feature $h_t^k(x)$,

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right), \tag{2}$$

where r_t is a weighted pairwise mis-ordering rate

$$r_t = \sum_{x_0}\sum_{x_1} C_t(x_0,x_1)\left[h_t^k(x_1) - h_t^k(x_0)\right]. \tag{3}$$

RankBoost and its variants constitute a family of widely used algorithms for ranking problems, because they can offer good generalization performance by jointly selecting the ranking features and optimizing their combination weights. Yet, they present a limitation that every unranked sample are treated equally, no matter where it is localized in the ranked list. In both learning and prediction stage, RankBoost must evaluate every weak feature h_t on the entire collection, whereas users tends to focus more on top ranked results in a typical ranking and retrieval scenarios. Therefore, its ranking process will become unnecessarily prohibitive if similar resources are spent in the top samples as the data residing at bottom. An efficient ranking process, on the other hand, should exploit fewer resources for as many irrelevant data as possible.

Imbalanced RankBoost

1. Input: training set X, a pool $\mathcal{H}$ of ranking features $h^k(x) \in [0,1]$
2. For $t = 1, \ldots, T$
   (a) Select h_t^k and θ_t^n that achieves the minimal value L_t' in the loss matrix $\mathbf{L_t}$ computed by Eq. 17
   (b) Find the corresponding α_t for the optimal L_t'
   (c) Go to Step 3 if L_t converges, i.e., $|L_t' - L_{t-1}'| < \epsilon$
   (d) Update ranking function $f_t \leftarrow f_{t-1} + \alpha_t h_t^k I[f_{t-1} \geq \theta_t]$
3. Output: final ranking $F(x) = \sum_{t=0}^{T} \alpha_t h_t(x) I\left[\sum_{q=0}^{t-1} \alpha_q h_q(x) \geq \theta_t\right]$

Figure 3. Outline of Imbalanced RankBoost.

4. Imbalanced RankBoost

In order to gradually migrate computational resources to more relevant data in the learning process, we develop a rank learning algorithm called Imbalanced RankBoost, which augments RankBoost with an iterative thresholding scheme in a unified loss optimization framework. In each iteration t, the proposed approach can learn, select and combine a set of weak ranking features by choosing ranking features h_t^k and weights α_t. Simultaneously, it is also able to identify a monotonically increasing cutoff threshold θ_t, and automatically truncate ranking feature computation for the lower-ranked examples. This enables the learning process to iteratively assign additional weak features to re-order the top ranked data, with the lower ranked samples being intact. To implement such a thresholding approach, we introduce a new identification function for each ranking feature, and this leads to the following ranking function,

$$\begin{aligned} F(x) &= \sum_{t=0}^{T} \alpha_t h_t^k(x) I[f_{t-1}(x) \geq \theta_t] \\ &= \sum_{t=0}^{T} \alpha_t h_t^k(x) I\left[\sum_{q=0}^{t-1} \alpha_q h_q^k(x) \geq \theta_t\right], \theta_t \geq \theta_{t-1}. \end{aligned} \tag{4}$$

Figure 3 presents the Imbalanced RankBoost algorithm. The initialization step computes α_0 and h^0 in the same way as RankBoost, because there are no thresholds involved in the first iteration. θ_0 is set to $\alpha_0 \cdot \min_{x_n} h^0(x_n)$, which trivially takes all the data into account. After the first iteration, the algorithm starts to diverge from RankBoost due to the more aggressive cut-off threshold. At each iteration t, the loss function L_t is estimated for all possible values of $h^k(x)$ and θ, and both of them will be jointly optimized (Section

4.2). Ideally, the optimal threshold θ_t is expected to skip the prediction of irrelevant data as early as possible, while keep most of relevant samples in the loop. At the same time, the optimal α_t can be obtained in a closed form (Section 4.1). These estimation steps will iterate until the loss function converges, or a certain iteration T is reached. Finally, the ranking function $F(x)$ is a linear combination of selected ranking features h_t, weighted by α_t together with an pruning indicator function against the cut-off threshold θ_t.

With a monotonically increasing cutoff threshold, our algorithm can iteratively incorporate more ranking features on a smaller set of higher ranked examples. While generally beneficial, such a learning process could occasionally become counterproductive when the cut-off thresholds increases too aggressively without any constraints. For instance, if an large θ_t is unexpectedly selected an an outlier, there will be only very few examples left to be continuously updated in the following runs, and it will lead to pre-mature optimization process. To mitigate this overfitting issue, we can introduce an additional regularization term $\Omega(\theta)$ to the original RankBoost exponential loss function when learning the threshold θ_t,

$$L = \sum_{x_1,x_0} \exp\left[-\left(F(x_0) - F(x_1)\right)\right] + \lambda\Omega(\theta). \quad (5)$$

At iteration t, the loss function L_t can be rewritten as,

$$L_t = \sum_{x_1,x_0} Z_t C_t(x_0,x_1) \exp[-A(x_1,x_0)] + \lambda\Omega(\theta), \quad (6)$$

where Z_t and C_t are similarly defined as Eq. 1 to encode the normalized distribution at iteration t, λ is the regularization factor, $\Omega(\theta)$ is the regularization term, and $A(x_1,x_0)$ is defined as

$$\begin{aligned} A(x_1,x_0) = {} & h_t^k(x_1)\, I\left[f_{t-1}(x_1) \geq \theta_t\right] \\ & - h_t^k(x_0)\, I\left[f_{t-1}(x_0) \geq \theta_t\right]. \end{aligned} \quad (7)$$

The regularization term $\Omega(\theta)$ should be designed based on the following principles, i.e., it should penalize larger distance between the neighbor thresholds θ_t and θ_{t-1} under the assumption that $\theta_t \geq \theta_{t-1} \forall t$. We proposed three following regularizers $\Omega(\theta)$ consistent with such principle, including two quadratic and one exponential,

$$\Omega(\theta_t) = \begin{cases} \sum_{l=1}^{t} (\theta_l - \theta_{l-1})^2 \\ \sum_{l=1}^{t} \exp(\theta_l - \theta_{l-1}) \\ \sum_{l=1}^{t} \left(\theta_l - \min_{f_{l-1}} (f_{l-1}(x) > \theta_{l-1})\right)^2 \end{cases}. \quad (8)$$

In the rest of this section, we describe in detail how the parameters h_t^k, α_t and θ_t are computed at each iteration.

4.1. Learning combination weights α_t

In the following discussions, we study the optimization approaches for the combination weights α_t. Since the regularization term $\lambda\Omega(\theta)$ does not involve α_t, we discard $\lambda\Omega(\theta)$ for the sake of simplicity. To derive a closed form solution for the optimal α_t, we relax the exponential loss function to its upper bound by using a similar exponential inequality adopted in RankBoost,

$$L_t \leq Z_t \left[\left(\frac{1-r_t}{2}\right) e^{\alpha_t} + \left(\frac{1+r_t}{2}\right) e^{-\alpha_t} \right], \quad (9)$$

where r_t is a weighted pairwise mis-ordering rate

$$r_t = \sum_{x_1,x_0} C_t(x_1,x_0)\, A(x_1,x_0). \quad (10)$$

By simply setting the derivative of L_t to 0, we can find a closed form solution for the optimal α_t,

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+r_t}{1-r_t}\right). \quad (11)$$

This minimizer α_t bears a similar form as the minimizer in Eq. 2, but in this case r_t is derived from a "pruned" set of weak ranking functions. Substituting α_t back to the loss function L yields the approximate upper bound as follows,

$$L_t \leq L_t' = Z_t \sqrt{1 - r_t^2} \;+\; \lambda\Omega(\theta). \quad (12)$$

4.2. Learning threshold θ_t and ranking features h_t^k

By minimizing the approximate upper bound L_t' in Eq. 12, we can learn the optimal threshold θ_t and ranking feature h_t together in a joint process. Because the ranking follows the bipartite structure, we can decompose $C_t(x_1,x_0)$ into two independent multipliers,

$$\begin{aligned} C_t(x_1,x_0) &= \frac{\exp - (f_{t-1}(x_0))}{Z_{t1}} \cdot \frac{\exp(f_{t-1}(x_1))}{Z_{t0}} \\ &= v_t(x_1) \cdot v_t(x_0). \end{aligned} \quad (13)$$

This then converts r_t to an aggregate form that can be computed more efficiently,

$$r_t = \sum_{n=1}^{N} s(x_n) v_t(x_n) \left[h_t^k(x_n)\, I\left[f_{t-1}(x_n) \geq \theta_t\right]\right], \quad (14)$$

$$s(x_n) = \begin{cases} +1 & if \;\; x_n \in X_1 \\ -1 & if \;\; x_n \in X_0. \end{cases} \quad (15)$$

However, because θ_t is embedded in the threshold function, it is difficult to directly optimize L_t' by using any gradient-descent based methods. It is also impossible to enumerate all possible values of θ_t which is infinite. As

an alternative, we describe an efficient method for exact optimization as follows. First, we reduce the search space of θ_t from $\mathbb{R}$ to $\mathbb{F} = [f_{t-1}(x_1), \ldots, f_{t-1}(x_N)]$. This reduction is sensible, because the thresholds θ_t only appear in the indicator function $I[\cdot]$) when only the boundary scenarios are required to be considered, and the regularization term $\Omega(\theta_t)$ is a monotonically increasing function when θ_t is constrained to be larger than θ_{t-1}.

The next step is to sort the training examples x_n with a descending order of the latest ranking function $f_{t-1}(x_n)$, which constitute the possible choices for θ_t. All the possible values of upper bound L'_t can be now organized into a matrix $\mathbf{L_t}$ as shown in Figure 4, of which the columns correspond to base models h, and the rows correspond to training examples. Based on the definition of r_t^k, we can use the following iterative update rules to efficiently compute the values of $r_t^k(x_n)$,

$$r_t^k(x_n) = r_t^k(x_{n-1}) + s(x_n)\, v_t(x_n)\, h_t^k(x_n). \quad (16)$$

The values $r_t^k(x_n)$ can then be substituted back into Eq. 12 to compute each loss entry $l_t^k(x_n)$.

$$l_t^k(x_n) = Z_t \sqrt{1 - \left(r_t^k(x_n)\right)^2} \; + \; \lambda \Omega(\theta_t^n). \quad (17)$$

Figure 4 shows an example of the matrix $\mathbf{L_t}$ and the joint optimization steps of θ_t and h_t. Without the pre-sorting approach presented in Eq. 16, it would require a computational time of $O(KN)$ to build the loss matrix with a direct computation of r_t^k in Equation 14. However, our pre-sorting method can significantly improve the computational efficiency. Since the sorting operation can be performed in $O(N \log N)$, the time complexity of each learning round can be reduced from $O(K^2N)$ to $O(N \log N) + O(KN)$. As K is typically much larger than $\log N$, Imbalanced RankBoost can learned in a similar duration as the standard RankBoost, while it is able to significantly reduce the prediction time with aid of the increasing cut-off thresholds.

The advantage of Imbalanced RankBoost consists in a more efficient ranking process without degrading, yet even improving the output performance. To better understand how the newly introduced cut-off thresholds θ can speed up the ranking process, we provide an illustrative example in Figure 5 to show how the final ranking function $F(x_n)$ is applied to each instance for both RankBoost and Imbalanced RankBoost. Initially, both methods evaluate the ranking score to be $f_0(x_n)$. But in the successive steps, RankBoost will start to merge a weighted ranking feature h_t^k with the existing ranking function for the entire collection, but Imbalanced RankBoost will only do so when the ranking function $f_{t-1}(x_n)$ exceeds the threshold θ_t. As can be observed from Figure 5, the number of ranking features evaluated by RankBoost is a constant no matter where the examples are positioned. Imbalanced RankBoost, on the other hand, does not compute the ranking feature h_t^k unless the ranking function exceeds the threshold θ_t (e.g., the examples x_1 to x_j in our case). As a result, it deploys fewer features to order the examples with lower ranks, which also have decreasing values from a user perspective.

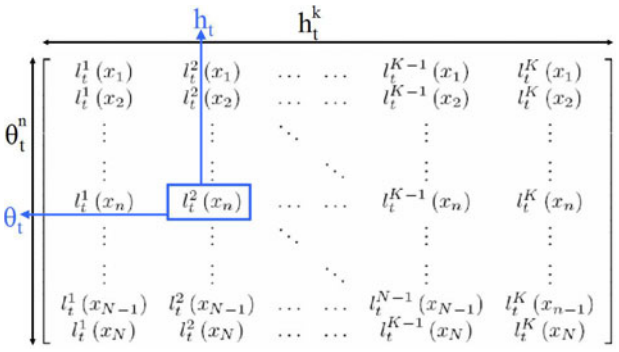

Figure 4. Matrix $\mathbf{L_t}$ representing the possible values of the loss function at step t, ordered by thresholds θ_t^n (columns) and ranking features h_t^k (rows). Finding the minimum value of $\mathbf{L_t}$ automatically enforces the choice of θ_t and h_t

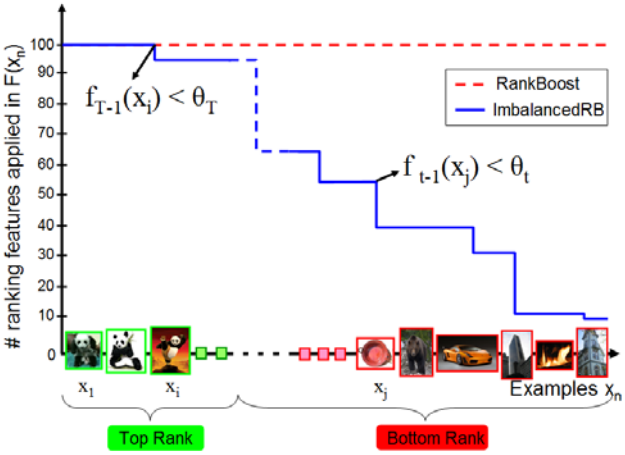

Figure 5. Illustrative example of RankBoost vs. Imbalanced RankBoost. RankBoost applied the same number of ranking features to both the top and bottom ranked examples (red dash line). Imbalanced RankBoost, instead, applied the learned thresholds θ_t to prune ranking features on the bottom rank data (blue solid line), while preserving most of the features for the top ranked ones. This thus resulting in a more efficient ranking process.

5. Experimental Results

In this section, we present the a series of experimental results on the TRECVID high-level feature extraction benchmark [11] in order to compare the performance between the proposed approach and its baseline algorithm.

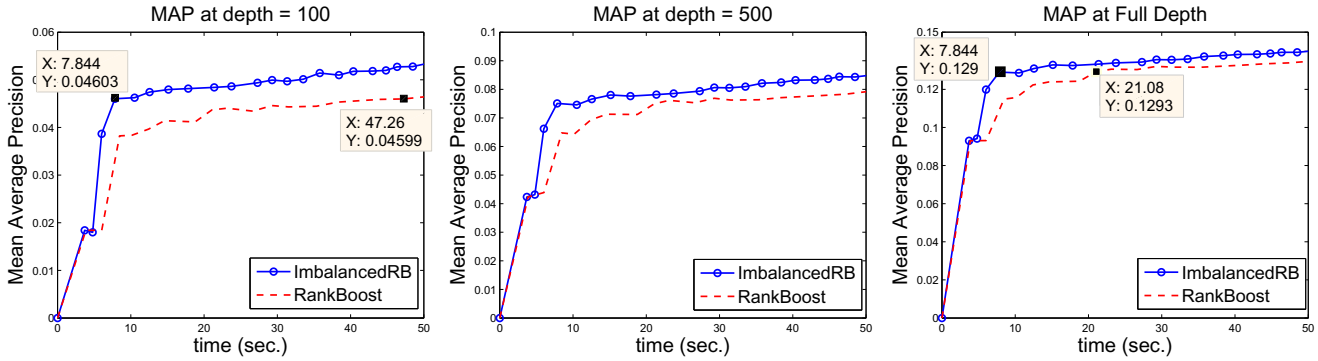

Figure 6. Mean average precision computed as function of the ranking time at top ranked 100 samples (left), top ranked 500 samples (center), full depth (right). Comparison of Imbalanced RankBoost and original RankBoost on the TRECVID 2007 benchmark.

5.1. Experimental setting

We evaluated the performance of Imbalanced RankBoost on the TRECVID 2007 high level feature extraction benchmark [11], which is currently the largest annotated video collection publicly available. It consists of 100 hours of news magazine, science news, news reports, documentaries, educational programming, and archival videos. We obtained a total of 52,347 keyframes of the development set from [8] and [1]. Each keyframe is annotated with the 20 semantic concepts officially provided in the TRECVID 2008 high level features extraction task (mainly in the categories of people, places, things, activities). On average, the ratio of the number of irrelevant to relevant keyframes is 49.47, with a peak of 385.75 for the 'Airplane flying' concept. This demonstrates a high data imbalance in the collection. In our experiments, we randomly split the development collection into the following: 50% as training data for learning ranking features, 20% as the held-out data for estimating boosting parameters, and the remaining 30% as testing data.

The input ranking features are generated by SVMs with RBF kernel on a number of bags of training data, while the SVM parameters are automatically selected through a 3-fold cross-validation. Each bag of training data was randomly sampled with a balanced number of positive and negative samples, where the maximum sizes are fixed to 800 examples. Moreover, each bag is generating from 1 out of 100 types of low-level visual features, including color histogram, color correlogram, color moment, wavelet texture and edge histogram computed at different granularities. We generated 2 weak models for each feature type, which results in 200 base models for each concept. Although these SVM-based ranking features are more robust than the most common base feature used in boosting, i.e., binary decision stumps on a single visual feature, they can still be considered "weak" due to the small data sampling rate, and simple feature space. In all the experiments, the outputs of above models is normalized to the range [-1,1]. For Imbalanced RankBoost, unless stated otherwise, we use the squared distance $\sum_{l=1}^{t} (\theta_l - \theta_{l-1})^2$ as the regularization term, and $\lambda = 10^5$ as the regularization factor.

We evaluate the effectiveness for ranking systems using mean average precision (AP). For each concept, let TP be the total number of relevant samples in the collection (which contains a total N samples). Let TP_d be the number of relevant samples found in the top d ranked samples returned by the system. Let $I_d = 1$ if the d^{th} sample in the ranked list is relevant and 0 otherwise. The average precision (AP) is then defined as $\frac{1}{TP} \sum_{d=1}^{N} \frac{TP_d}{d} I_d$. The AP can be computed at different depths by changing the value of N. The mean of average precision over all the concepts is defined mean average precision (MAP).

5.2. Results

Figure 6 compares the mean average precision of Imbalanced RankBoost against RankBoost as a function of ranking time for the testing collections. The performance of both algorithms is evaluated on several depths including 100 keyframes, 500 keyframes, and full depth. [3] Measuring performances at intermediate depths is important in light of the fact that users typically focus on a limited number of top ranked examples. The running times were empirically measured during the experiments. We choose to limit all the figures up to 50 seconds because we observed that both algorithms tend to converge to its asymptote performance afterwards.

From Figure 6, we can find that Imbalanced RankBoost outperforms the original RankBoost in all three cases. One interesting observation is that the improvement becomes more evident when the MAPs are computed at smaller depths. For example, with a depth of 100, it registers up to a 6-fold speed-up in the ranking process when both of them reach a mean average precision of 0.046. When the perfor-

[3] The concept of "depth refers to the number of instances which are considered in the evaluations. For example, Average Precision computed at depth 100 means that the AP was computed only on the top 100 instances ranked by the system.

mance are evaluated at the full depth, it can still achieve the same levels of performance with a 3-fold speedup. From another perspective, instead of degrading the ranking performance, the speedup of ranking process of Imbalanced RankBoost also brings additional improvement gain by focusing more on relevant examples, which achieves an relatively improvement ranged from 7% to 21% at a fixed time (50 seconds). These observations demonstrate that the proposed Imbalanced RankBoost algorithm can effectively redistribute the computational resources among relevant and irrelevant examples, and dedicate more accurate examination to the most valuable results.

To further analyze the properties of Imbalanced RankBoost, Figure 7 depicts the performance comparison on a per concept basis. The average precise at full depth is computed at the ranking time $t = 7$ seconds, where the difference between the two algorithms is maximum. For 17 out of 20 concepts, Imbalanced RankBoost presents an either better or equivalent average precision scores, which is statistically significant. Figure 8 shows the performance of four representative concepts including "Cityspace", "Classroom", "Hand" and "Flower". For the first three concepts, the performance gain between Imbalanced RankBoost and RankBoost is quite considerable. On the other hand, for the concept of "Flower", the thresholding process in Imbalanced RankBoost becomes too aggressive and thus degrade the ranking performance after incorporating more base models.

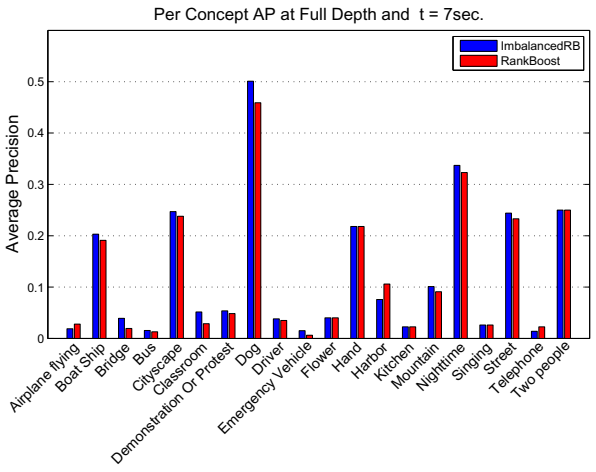

Figure 7. Average precision per concept computed at full depth at $t = 7$ seconds. Comparison of Imbalanced RankBoost and original RankBoost methods on the TRECVID 2007 benchmark.

Finally, Figure 9 investigates the ranking performances when the different regularization terms are applied with a range of regularization factors λ. It shows that the ranking performances are not sensitive to one or another regularization terms, which verifies the robustness of the proposed algorithm over choices of regularizers.

6. Conclusion and future work

In this paper, we proposed a new rank learning algorithm called Imbalanced RankBoost, which combines RankBoost and iterative thresholding into a unified loss optimization framework. Unlike the RankBoost algorithm which has equal treatment for all the data, the proposed approach provides a more efficient ranking process by applying a thresholding process to automatically emphasize top ranked examples and truncate ranking feature computation for less important bottom-ranked ones. This property makes Imbalanced RankBoost particularly suitable for the ranking problems on large-scale data collections with a significant imbalance between relevant and irrelevant instances. Experiments on the TRECVID 2007 high-level feature detection benchmark showed that the proposed approach can outperform RankBoost in terms of both ranking effectiveness and efficiency. It achieves an up to 21% improvement in terms of mean average precision, or equivalently, a 6-fold speedup in the ranking process. Future directions involve directly incorporating the processing time of ranking features in the learning process, so that faster ranking features will be selected even when they produce a slightly inferior loss reduction than slower features.

References

[1] S. Ayache and G. Quenot. Evaluation of active learning strategies for video indexing. In *International Workshop on Content-Based Multimedia Indexing(CBMI)*, 2007. 6

[2] E. Bruno, J. Kludas, and S. Marchand-Maillet. Combining multimodal preferences for multimedia information retrieval. In *MIR '07: Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 71–78, 2007. 1

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005. 2

[4] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2006. 1, 2

[5] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting, 2003. 2

[6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003. 1, 2, 3

[7] H. Guo. Learning from imbalanced data sets with boosting and data generation: The databoost-im approach. *SIGKDD Explorations*, 6:30–39, 2004. 2

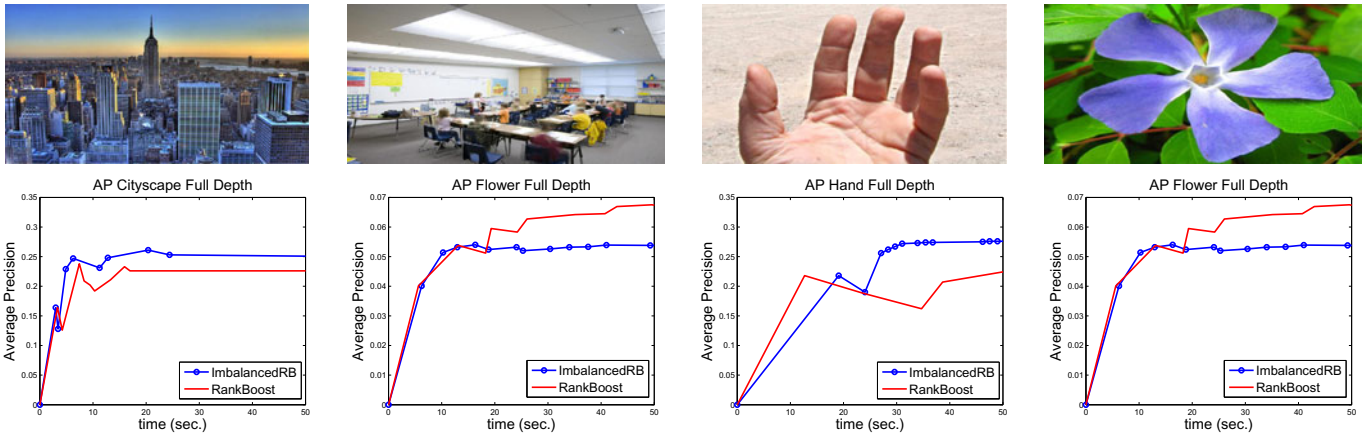

Figure 8. Average precision for single concepts computed as function of the ranking time at full depth on the TRECVID 2007 benchmark. Imbalanced RankBoost performs better for 'Citycape', 'Classroom' and and 'Hand'(a, b and c), worse on 'Flower' (d).

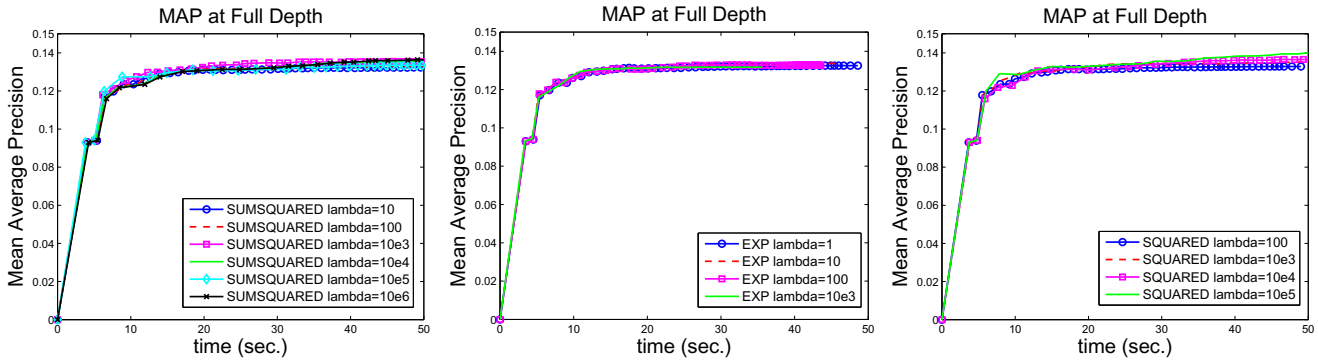

Figure 9. Analysis of the influence of $\Omega(\theta)$ to the mean average precision computed as function of the ranking time at full depth. The choices of λ and $\Omega(\theta)$ do not affect significantly the performances of the algorithm.

[8] Z. Liu, D. Gibbon, E. Zavesky, B. Shahraray, and P. Haffner. A fast, comprehensive shot boundary determination system. In *ICME*, pages 1487–1490. IEEE, 2007. 6

[9] H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 609–619, New York, NY, USA, 2007. ACM. 2

[10] S. Rajaram, C. D. Dagli, N. Petrovic, and T. S. Huang. Diverse active ranking for multimedia search. In *SLAM07*, pages 1–8, 2007. 1, 2

[11] A. Smeaton and P. Over. TRECVID: Benchmarking the effectiveness of information retrieval tasks on digital video. In *Proc. of the Intl. Conf. on Image and Video Retrieval*, pages 19–27, 2003. 5, 6

[12] A. Spink, D. Wolfram, B. J. Jansen, and T. Saracevic. *Searching the web: the public and their queries*, volume 52(3), pages 226–234. 2001. 1

[13] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002. 2

[14] B. X. Wang and N. Japkowicz. Boosting support vector machines for imbalanced data sets. In *ISMIS*, pages 38–47, 2008. 2

[15] D. Wang, J. Li, and B. Zhang. Relay boost fusion for learning rare concepts in multimedia. pages 271–280, 2006. 1, 2

[16] H. Wu, X. Liu, and G. Doretto. Face alignment via boosted ranking model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008. 2

[17] R. Yan and A. G. Hauptmann. Efficient margin-based rank learning algorithms for information retrieval. In *CIVR06*, pages 113–122, 2006. 1